# Measuring Software Engineering Report

Aaron Readman

Student ID: 19334583

## Contents

## Introduction

In this report I shall discuss different ways software engineering can be measured. I will also provide examples of different platforms that can be used to gather information as well as the algorithms one can use when processing the data. I will also discuss the ethics of these things and provide my opinion on the topic.

## How can Software Engineering be Measured?

This section describes various method for measuring software engineering. The three I have chosen are source lines of code, cycle and lead time and code coverage. Many of these metrics can apply to both the individual software engineer and to the performance of a team of software engineering.

### *Source Lines of Code*

One of the very first methods for measuring software engineering involved counting the lines of code produced (Fentan & Neil, 1999).

According to (Bhatt, Vinit, & Patel, 2012) there are several major flaws that accompany using lines of code as a main metric for measuring software engineering. Its possible that a developer might produce a large amount of code when working on a solution to a problem. Using lines of code as a metric for productivity would mean that the developer has been productive. However if another developer working on that same problem produces a solution using less code then they might be considered less productive when compared to the first developer. The second developer could be significantly more skilled than the first developer and thus will be able to solve the problem with less code. Using source lines of code does not take this into account as it ignores the functionality of solutions.

Another issue with source lines of code that (Bhatt, Vinit, & Patel, 2012) show is that this metric does not take into account the different between programming languages. A high-level language such as java and a low-level language like assembly would require different amounts of effort when working on a solution and the size of lines of code would be different.

Finally, it is very easy for a developer to take advantage of this metric. They could deliberately add more complexity to the code. Using Lines of code as a metric for measurement incentivises developers to write more even if it is not necessary. This causes the program to become larger, more difficult to maintain and introduces potential for more bugs to appear (Bhatt, Vinit, & Patel, 2012).

## *Cycle Time And Lead Time*

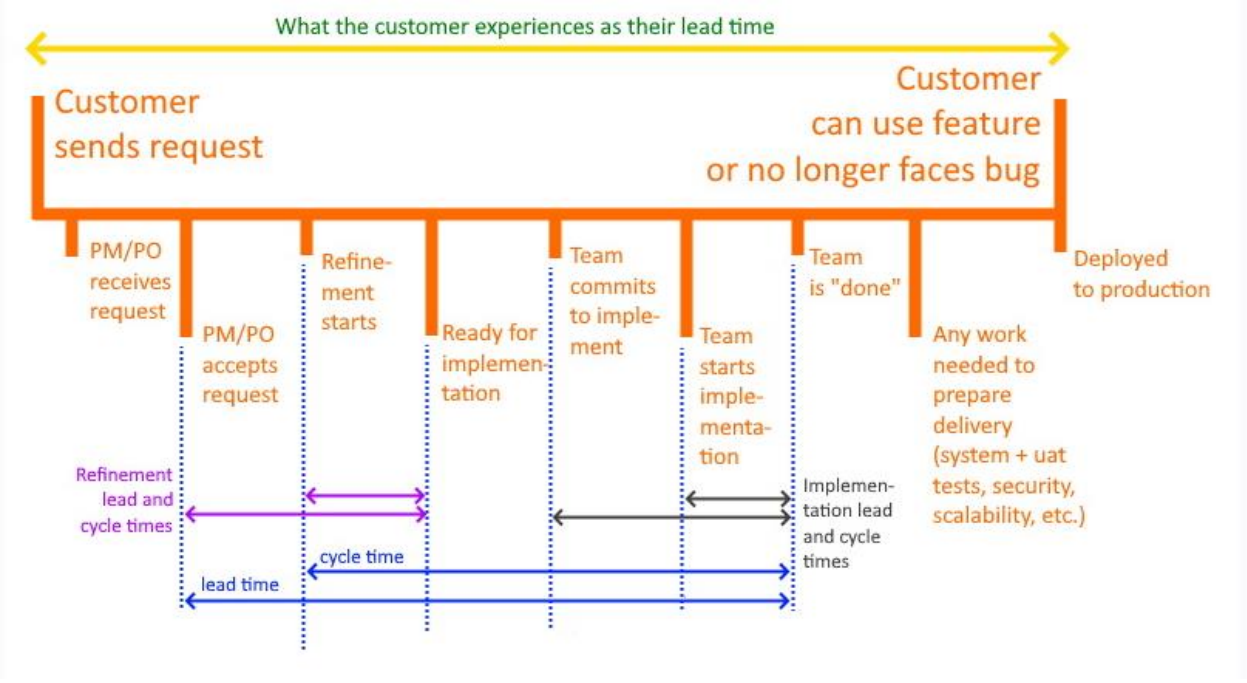


*Image taken from* (Digite, 2021).

A metric for measuring team performance for software engineering is cycle and lead time as described in (Digite, 2021). Both these metrics are used in agile development. Lead time is the time it takes between a user making a request to a team and them actually fulfilling that

request for the user so they can make use of it and gain value from it. The components of lead time are from when the user makes the request to when the team begins working on the project and then from that point to them delivering to the user. Cycle time only takes into account the time from when the team starts working on the request and ignores the time the request spent in their backlog.

The data returned by lead and cycle time will inform a manager or data analyst whether the times are stable or volatile (Digite, 2021). A stable cycle time will mean the team has not encountered major issues and are working consistently. An unstable cycle time would mean that the team is facing difficulties. This could be due to a number of reasons such as needing to train new members or suffering from system failures.

According to (Agile Academy, 2021) it is better for the cycle and lead time to be short rather than long as users benefit from getting results quickly. A manager might use this metric to see if the team has too long of a lead or cycle time.

### Code Coverage

Another metric for measuring the quality of work being produced in software engineering is code coverage. (Bose, 2020) states *that "A high percentage of code coverage results in lower chances of unidentified bugs.".* Bose describes multiple different types of code coverage such as statement coverage which checks that all executable statement in the code base is executed once or more. Code coverage is beneficial for developers because it allows them to measure how much of their code is being used as well as remove redundant code. It also encourages a high degree of unit testing. Developers with a high code coverage should in theory have less unidentified bugs and therefore they are producing higher quality work so using code coverage as a measurement tool for software engineering helps measure the quality of work produced.

## What Platforms can be used to gather and process data

# GitHub Rest API

GitHub is a version control service created in 2008. (Traversa, 2018). While many people use it as a a platform for simply storing their work, the service also collects and stores data than can be accessed using something called the GitHub Rest API (Github, 2021). GitHub will record various different pieces of information about the work users do as well as about the users themselves and make them publicly accessible to the outside world. For example, using Github Rest API , one could get access to every public repo, see the users that have contributed to those repos and even go further and count how many contributions each user has made (Github, 2021).

# Codacy

Codacy is an analysis software that analyses the quality of the work as it is being produced as shown in their webpage (Codacy, 2021). The site outlines how it can be used on any git based provider like GitHub, BitBucket or GitLab. Examples of code analysis it does is things such as code coverage, lines of code covered, errors and the security of the code. This platform provides an easy way to view the quality of code produced and see just how comprehensive any unit tests designed are. Below is an example of the information it provides.

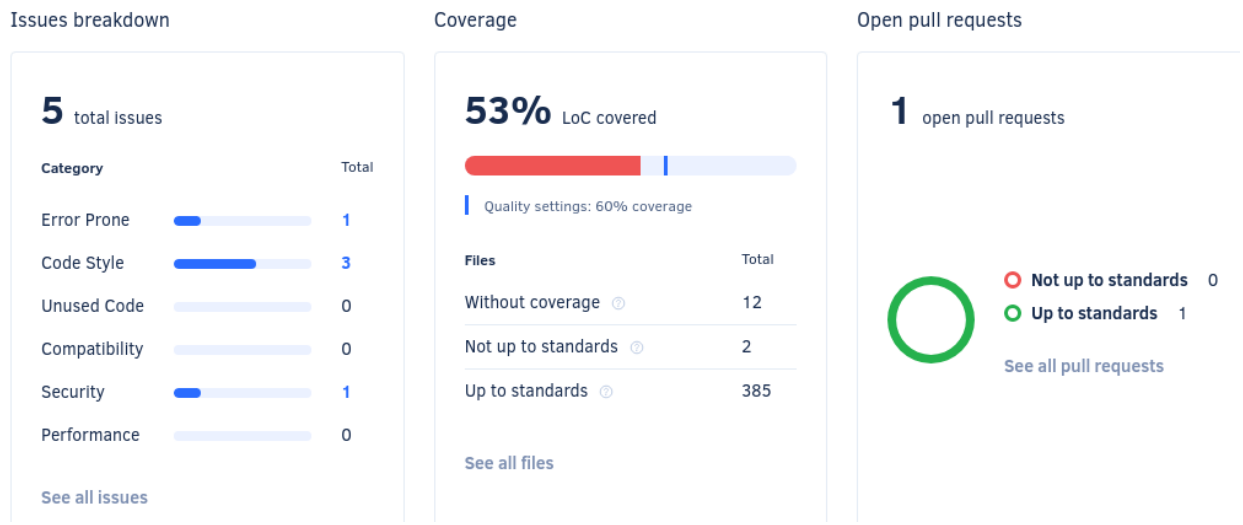


Image taken from (Codacy, 2021)

## WayDev

WayDev is another software engineering measurement platform that can measure the work of both teams and the individuals in that team (Waydev, 2021). It provides the ability to measure various aspects of a developers work such as the following list

- Throughput;
- Productive Throughput;
- Efficiency;
- Technical Debt;
- Days Active;
- Commits;
- Low-Risk Commits;
- Medium-Risk Commits;
- High-Risk Commits;
- New Work;
- Legacy Refactor;
- Help Others;
- Churn

*Information obtained from Wayde website* (Waydev, 2021)

As can be seen above, they provide a variety of metrics to evaluate a developers efficiency. They look at the number of commits they commit as well as the riskiness of those commits. They also view the churn of the developer, the days they are active and the technological debt they enter into. The platform also provides team based evluatiions for software engineering. They measure things such as the cycle and lead time of the team, the cost of the project, the time taken and other metrics that help with analysing software engineering.

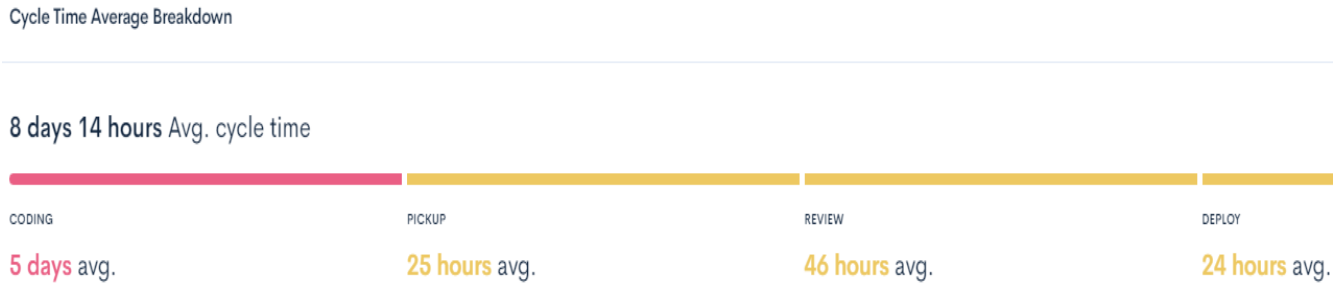


*Image taken from* (waydev, 2021)

## What algorithms can we use

There are many algorithms and approaches in place that exist for the purpose of looking at data. This section will focus the use of machine learning

### Unsupervised Learning

IBM (IBM , 2021) states that unsupervised machine learning will use algorithms to analyse data without human intervention and that *"Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis".* Unsupervised machine learning algorithms should be able to analyses the software engineering measurement data captured on platforms and then spot patterns or similarities that exist within the data (IBM , 2021). There is no output data required for this to work as the unsupervised learning only requires input data (Javatpoint, 2021).

There are several algorithms that can do this and one example is K means Clustering:

#### *K means Clustering*

K means clustering groups data points based on how similar they are to one another (Gadde, 2021). This algorithm as described by (Gadde, 2021) begins by first having a value K which indicates the number of clusters it needs to have. Then a center point or "centroid" is randomly selected for each cluster. After that, the distance between the central point and each data point is calculated. Clusters are then created around each centroid. Then a new centroid is chosen for each centroid that best fits each cluster. This will repeat until no new centroids need to be chosen as they are each the best fit available. Below are two images that are made to easily show out clusters work. The first image is a graph without clusters and the 2nd is the graph with clusters. The centroid points are visible in the 2nd graph.

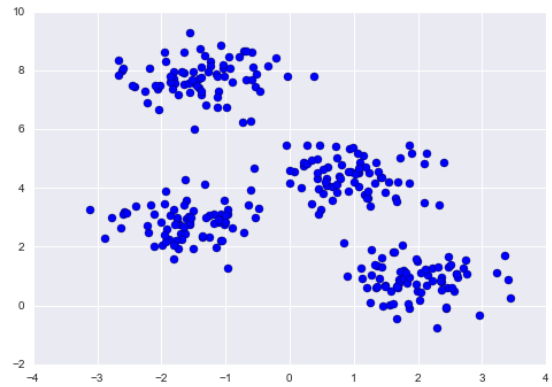

*Image taken from* (VanderPlas, 2021)

*Image taken from* (VanderPlas, 2021)

## Supervised Learning

Supervised learning is another machine learning method that can be used to deal with the data generated from measuring software engineers. Unlike unsupervised learning which only deals with inputs, supervised learning involves having correct outputs (IBM, 2021). IBM describes supervised learning as teaching a model to produce a desired output. It will be given a training set of data which includes inputs and outputs and the algorithm will check how accurate it is and improve over time using this training data. An example of where supervised learning is used is Linear Regression.

### *Simple Linear Regression*

In linear regression, the algorithm attempts to plot the best fit line on a graph. This line should help someone see how a dependent variable and an independent variable will interact. One could use Linear regression to make plans about future outcomes when given an input.

According to (Deepanshi, 2021) the equation of simple linear regression is y=b0+b1x where y is the dependent variable, x is the independent variable, b0 is the intercept on the y axis and b1 is the slope of the line. A machine learning algorithm would use the input and output data provided as well as this formula and attempt to produce the best fit line by minimising the error between the values it produces and the correct outputs it has been given. An example of a best fit line is shown in the image below.

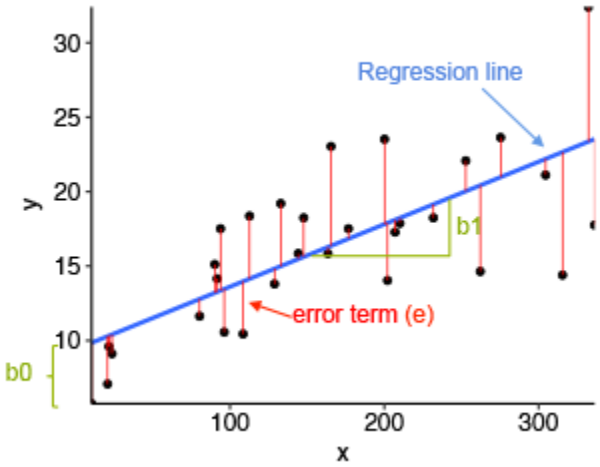


Image taken from (Deepanshi, 2021)

## Ethical Issues

I believe that for the most part, collecting data about software engineers using various measurements, platforms and algorithms is a good thing. Having these measurements on hands will allow managers to properly evaluate their employees. For example, platform Waydev provides various metrics to measure software engineers (Waydev, 2021). A manager could use this information combined with different algorithms and tell if an employee is having difficulty or is doing a good job. However even with all these benefits, there are still certain issues that might occur. Measuring developers might be an invasion of their privacy and these measurements might not collect data relevant to the work they are doing.

It's also possible that this data could be used incorrectly or without regard to nuance. An employee might be punished or let go if their measurements are poor. This might sound reasonable but its possible there were extenuating circumstances impacting their performance. Managers should use these measurements as a helpful tool to assist them with their work but shouldn't completely rely on them.

There has been legislation introduced to prevent situations where data collected is misued as seen in the General Data Protection Regulation(GDPR) (THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION, 2016). It ensures that data is correctly stored, kept secure and used for the correct purposes.

I think so long as anyone measuring software engineering complies with the GDPR and maintains transparency on the data they are collecting then it should not be too much of an issue.

# References


Agile Academy. (2021). *Agile Academy lead-time-vs-cycle-time/*. Retrieved from Agile Academy : https://www.agile-academy.com/en/agile-dictionary/lead-time-vs-cycle-time/

Bhatt, K., Vinit, T., & Patel, P. (2012, May 5). *Research Gate Analysis_Of_Source_Lines_Of_CodeSLOC_Metric.* Retrieved from Research Gate: https://www.researchgate.net/publication/281840565_Analysis_Of_Source_Lines_Of_CodeSLOC_Metric

Bose, S. (2020, March 26). *Browserstack code-coverage-vs-test-coverage*. Retrieved from www.browserstack: https://www.browserstack.com/guide/code-coverage-vs-test-coverage

Codacy. (2021). *codacy-quickstart*. Retrieved from docs.codacy.com: https://docs.codacy.com/getting-started/codacy-quickstart/

Codacy. (2021). *Repository-dashboard/*. Retrieved from docs.codacy.com: https://docs.codacy.com/repositories/repository-dashboard/

Deepanshi, D. (2021, May 25). *analyticsvidhya all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/*. Retrieved from analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/

Digite. (2021). *digite /lead-time-cycle-time/*. Retrieved from Digite: https://www.digite.com/agile/lead-time-cycle-time/

Fentan, N. E., & Neil, M. (1999). *Software metrics: successes, failures and new directions.* London: Journal of Systems and Software.

Gadde, M. (2021, February 22). *simple-explanation-to-understand-k-means-clustering/*. Retrieved from analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/02/simple-explanation-to-understand-k-means-clustering/

Github. (2021). *getting-started-with-the-rest-api*. Retrieved from github.com: https://docs.github.com/en/rest/guides/getting-started-with-the-rest-api

Github. (2021). *repos*. Retrieved from docs.github.com: https://docs.github.com/en/rest/reference/repos

IBM . (2021). *IBM unsupervised-learning*. Retrieved from ibm: https://www.ibm.com/topics/unsupervised-learning

IBM. (2021). *supervised-learning*. Retrieved from ibm: https://www.ibm.com/topics/supervised-learning

Javatpoint. (2021). *javatpoint*. Retrieved from unsupervised-machine-learning: https://www.javatpoint.com/unsupervised-machine-learning

THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION. (2016, April 26). *General Data Protection Regulation.* Retrieved from gdpr-info: https://gdpr-info.eu/

Traversa, P. (2018, October 2). *github-a-brief-history-and-a-briefer-tutorial*. Retrieved from medium: https://medium.com/@peter.traversa/github-a-brief-history-and-a-briefer-tutorial-576471e44200

VanderPlas, J. (2021). *jakevdp.github PythonDataScienceHandbook*. Retrieved from jakevdp.github: https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html

waydev. (2021). *product-live-demo*. Retrieved from Waydeb: https://waydev.co/product-live-demo/

Waydev. (2021). *Waydev*. Retrieved from Waydev features: https://waydev.co/features/

Waydev. (2021). *Waydev developer-progress*. Retrieved from Waydev: https://waydev.co/features/developer-progress/